Comparison of Methods for Probe Design

Noriyuki Hosaka
nori@hal.rcast.u-tokyo.ac.jp

Ken-ichi Kurata
kurata@hal.rcast.u-tokyo.ac.jp

Hiroshi Nakamura
nakamura@hal.rcast.u-tokyo.ac.jp

Research Center for Advanced Science and Technology, University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8904, Japan

Keywords: probe design, suffix tree

1 Introduction

Biology is unveiling the fundamental laws of creature at molecule level. Examining gene expression patterns is one of the approaches to the fundamental laws. Therefore, it is important to analyze the gene expression patterns, quickly, massively, and accurately. Currently, cDNA arrays and DNA chips are the solutions to the requirement. Many works have been carried out for the techniques of composing oligo DNA on the DNA chips and the cDNA arrays. However, methods designing oligo DNA sequences for such systems have not been well studied, although they have great influence on the accuracy of the systems. Hence, it's urgent work to establish effective and practical design methods.

The prior work [3] on the probe design for the known genome, has shown the direction of the study. To observe gene expressions accurately, the probes have to hybridize exclusively to their target genes. Consequently, each probe must be distinctive. It introduced a metric for distinctiveness, which is Unique Sequence length (USL) defined for each position in the genome. If once we have the USLT, which is a table of the USLs for the whole genome, we can obtain a proper set of probes for the genome by choosing low-USL regions of the genes. The method of [3] is quite fast, but still not fast enough to satisfy the need to design a set of probes for a large genome.

Then, we propose a new algorithmic method QT for constructing USLT. We also propose ST, a new approach for the probe design. ST directly produces a list of candidates for probes from a set of genes with out using USLT.

2 Algorithmic Methods for Probe Design

In this section, we present three algorithmic methods for probe design, Naive by [3], QT, and ST.

2.1 Naive

In Naive algorithm, USLT is obtained through two phases, which are the hashing phase and the base-to-base comparison phase. In the hashing phase, each position in the genes is classified with the key which is the sequence beginning from the position. The length of the key is arbitrary but was 7 in [3].

The second phase is the most time consuming part of the whole program. Neighboring bases of a position are compared one by one with neighboring bases of other positions having the same key, until mismatch is found. The length of the longest matching base sequence is the USL of that position. USLT is obtained by calculating USLs for all the positions of the genome.

Once the USLT is obtained, by sorting the USLT with its USL value, we have low USL value regions of genes are selected as candidates for the probes.

2.2 QT

QT(Quad-Tree) is an alternative algorithm for obtaining the USLT. It uses the hash as Naive does. The USL of the position is a sum of the left and right "partial" USL values. The following procedures are taken for both the left side and the right side of each position to obtain the "partial" USL value.

For all the positions with the same key, we construct one quad-tree (because alphabet $\Sigma = \{a, g, c, t\}$) according to left/right side bases of positions. The edges of the tree correspond to the bases. The tree grows until its each leaf node corresponds to the position by one-to-one. The depth of the leaf gives the "partial" USL value of the position.

2.3 ST

ST(Suffix Tree) is an algorithmic method to generate a set of probes for the genome. It constructs a suffix tree [2] for a set of genes. ST finds candidates which are similar to MUMs in MUMmer [1]. By traversing the suffix tree, they are found as nodes which are inner nodes(non-terminal nodes), nodes whose any child is a terminal node from the gene distinct from others, and nodes whose path-labels' left-side bases are different among leaves. For each gene, from all candidates region in the gene, it selects the shortest candidate of them, which is suitable as a probe.

3 Results

Table 1: Execution time of the programs in seconds

	E. coli(4.2MB)		*S. cerevisiae*(9.2MB)		*C. elegans*(21MB)	
	Naive	QT	Naive	QT	Naive	QT
Hash	13.10	13.09	28.58	28.49	66.88	67.06
USL-calc	2257.45	107.66	12357.50	272.02	96941.70	697.40
Selection	15.79	16.00	141.36	131.19	737.39	561.78
Total	2286.34	136.75	12527.44	431.70	97745.97	1326.24

We compared the performances of the Naive, QT algorithmic methods on Sun Ultra 450 workstation (300MHz) with 4 GB of memory. Input of the programs are FASTA formatted nucleotide sequences. We used the genomes of *E. coli*, *S. cerevisiae*, and *C. elegans*. The result is shown in Table 1. In the table, *Total* is the total execution time for each genome and algorithmic method. *Hash*, *USL-calc*, and *Selection*, are the breakdown of the total execution time, which represent the hashing phase, the phase of USL calculation for all positions in genome, and the phase of selecting a set of probes from the USLT by sorting the USLT with USL value, respectively. QT shows a significant improvement over Naive. The evaluation of ST is under progress.

References

[1] Delcher, A.L., Kasif, S., Fleischman, R.D., Peterson, J., White,O., and Salzberg, S.L., Alignment of whole genomes, *Nucleic Acid Research*, 27(11):2369–2376, 1999.

[2] Gusfield, D., *Algorithms on Strings, trees, and sequences*, Cambridge University Press, 1997.

[3] Kurata, K. and Nakamura, H., Novel method for primer/probe design and sequence analysis *Genome Informatics*, 11:331–332, 2000.